Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 782351, 10 pages
http://dx.doi.org/10.1155/2014/782351

Research Article

Comparing the Selected Transfer Functions and Local Optimization Methods for Neural Network Flood Runoff Forecast

Petr Maca, Pavel Pech, and Jiri Pavlasek

Department of Water Resources and Environmental Modeling, Faculty of Environmental Sciences, Czech University of Life Sciences Prague, Kamycka 1176, 165 21 Praha 6, Suchdol, Czech Republic

Correspondence should be addressed to Petr Maca; maca@fzp.czu.cz

Received 11 April 2014; Accepted 18 June 2014; Published 2 July 2014

Academic Editor: Jer-Guang Hsieh

The presented paper aims to analyze the influence of the selection of transfer function and training algorithms on neural network flood runoff forecast. Nine of the most significant flood events, caused by the extreme rainfall, were selected from 10 years of measurement on small headwater catchment in the Czech Republic, and flood runoff forecast was investigated using the extensive set of multilayer perceptrons with one hidden layer of neurons. The analyzed artificial neural network models with 11 different activation functions in hidden layer were trained using 7 local optimization algorithms. The results show that the Levenberg-Marquardt algorithm was superior compared to the remaining tested local optimization methods. When comparing the 11 nonlinear transfer functions, used in hidden layer neurons, the RootSig function was superior compared to the rest of analyzed activation functions.

1. Introduction

In recent three decades, the implementations of various models based on artificial neural networks (ANN) were intensively explored in hydrological engineering. The general reviews of ANNs modeling strategies and applications with the emphases on modeling of hydrological processes are presented in [1–3]. They confirm that the class of multilayer perceptron (MLP) [4, 5] belongs to the most frequently studied ANN's models in hydrological modeling [6–9].

The MLP forms the nonlinear data driven model. According to its architecture, it is a fully connected feed-forward network, which organizes the processing units (neurons) into the layers and allows the interconnections only between neurons in two following layers. As it was proved by [10], the MLP is the universal function approximator. This important property has been widely confirmed by many hydrological studies [11–14].

Despite the positive research results of a large number of studies on MLP runoff forecasting, there is a need for clear methodological recommendations of MLP transfer function selection [15, 22–24] combined together with the training method assessment and the implementation of new training method [8, 18, 19, 25].

Main aims of presented paper are (1) to analyze the hourly flood runoff forecast on small headwater catchment with MLP-ANN models, which are based on 12 different MLP's transfer functions following the work of [15, 24], (2) to compare the 7 local optimization algorithms [5, 17, 19], and finally (3) to evaluate the MLP performance with 4 selected model evaluation measures [26, 27].

2. Material and Methods

The tested runoff prediction using the MLP-ANN models uses the set of rainfall runoff data. The MLP-ANN implementation for runoff forecast generally consists of data preprocessing, model architecture selection, MLP training, and model validation. In this section, we give a very brief description of the MLP-ANN model architecture and tested optimization schemes and datasets.

Table 1: The implemented transfer functions [15]; a is the neuron's activation, y is the neuron output.

Function name	Transfer function	Derivatives of transfer function
Logistic sigmoid (LSi)	$y(a) = \frac{1}{1+\exp(-a)}$	$y'(a) = y(a)(1-y(a))$
Hyperbolic tangent (HT)	$y(a) = \tan h(a)$	$y'(a) = 1 - y(a)^2$
Linear function (LF)	$y(a) = a$	$y'(a) = 1$
Gaussian function (GF)	$y(a) = \exp(-a^2)$	$y'(a) = -0.5y(a)a$
Inverse abs (IA)	$y(a) = \frac{a}{1+\lvert a\rvert}$	$y'(a) = \frac{1}{(1+\lvert a\rvert)^2}$
LogLog (LL)	$y(a) = \exp(-\exp(-a))$	$y'(a) = \exp(-a)y(a)$
ClogLog (CL)	$y(a) = 1 - \exp(-\exp(a))$	$y'(a) = (1-y(a))\exp(a)$
ClogLogm (CLm)	$y(a) = 1 - 2\exp(-0.7\exp(a))$	$y'(a) = 1.4\exp(a)\exp(-0.7\exp(a))$
RootSig (RS)	$y(a) = \frac{a}{1+\sqrt{1+a^2}}$	$y'(a) = \frac{1}{\left(1+\sqrt{1+a^2}\right)\sqrt{1+a^2}}$
LogSig (LS)	$y(a) = \left(\frac{1}{1+\exp(-a)}\right)^2$	$y'(a) = \frac{2\exp(-a)}{(1+\exp(-a))^3}$
Sech (SF)	$y(a) = \frac{2}{\exp(a)+\exp(-a)}$	$y'(a) = -y(a)\tan h(a)$
Wave (WF)	$y(a) = (1-a^2)\exp(-a^2)$	$y'(a) = 2a\exp(-a^2)(-2+a^2)$

Table 2: The implemented local training gradient based methods.

Training method	References
Batch propagation (BP)	[4, 5, 16]
Batch backpropagation with regularization (BP_regul)	[4, 5, 16]
Levenberg-Marquardt (LM)	[4, 17, 18]
Scaled conjugate gradient, Perry (PER)	[19–21]
Scaled conjugate gradient, Polak (POL)	[19–21]
Scaled conjugate gradient, Hestenes (HEST)	[19–21]
Scaled conjugate gradient, Fletcher (FLET)	[19–21]

2.1. MLP-ANN Model. We analyzed the MLP model with one hidden layer. The similar ANN architecture was used in a large number of hydrologically oriented studies [18, 28–31]. The studied MLP models had in total three layers of neurons, the input layer, the hidden layer, and the output layer. As proved by Hornik et al. [10], this type of artificial neural network with sufficiently a large number of neurons in the second layer can approximate with desired precision any measurable functional relationship.

The implemented MLP-ANN models had a general form

$$R_f = v_0 + \sum_{j=1}^{N_{\text{hd}}} v_j f\left(w_{0j} + \sum_{i=1}^{N_{\text{in}}} w_{ji}x_i\right), \quad (1)$$

where the R_f is a network output, that is, flood runoff forecast for given time interval, x_i is network input for input layer neuron i, N_{in} is the number of MLP inputs, the w_{ji} is the weight of i input to j hidden layer neuron, $f()$ is the activation function constant for all hidden layer neurons, N_{hd} is the number of hidden neurons, v_j is the weight for output from hidden neuron j, and w_{0j}, v_0 are neuron biases [2–4, 18, 25, 31].

2.1.1. MLP-ANN Transfer Functions. The type of activation function together with network architecture influences the generalization of neural network. Imrie et al. [32] empirically confirmed that the transfer function bounding influences the ANN generalization and hydrological extreme simulations during runoff forecast. Following the work of [15], we implemented the 12 different types of transfer functions, and 11 of them were tested in hidden neuron layer of analyzed MLP-ANN models. Table 1 provides their list.

The activation functions type combined with specific type of training methods influences the average performance of leaning algorithm and computing time [15, 24]. For example, the Bishop [4] pointed out that the implementation of hyperbolic function speeds up the training process compared to the use of logistic sigmoid.

2.1.2. MLP-ANN Local Optimization Methods. We selected 7 gradient based local optimization methods. Table 2 shows their list together with their references. All MLP-ANN optimization was performed using the batch learning mode [4].

All tested gradient local search methods (except BP_regul) minimized the error function represented as a the sum of square of residuals

$$E_r = \frac{1}{2}\sum_{i=1}^{N} r_i^2, \quad (2)$$

and the residuals $r_i = R_o[i] - R_f[i]$ were defined as differences between observed R_o and computed $R_f[i]$ flood runoff.

The two first order local training methods are represented by the standard backpropagation and backpropagation with regularization term. Both backpropagation methods implement the following modification: constant learning rate and momentum parameter. The BP_regul used the regularization

term, which penalizes the size of estimated weights, and the error function is defined as

$$E = \beta E_r + \alpha \frac{1}{2} \sum_{l=1}^{Nw} w_l^2, \tag{3}$$

where the Nw is a total number of MLP-ANN weights w_l. The hyperparameters α and β were constant within the standard backpropagation with the regularization term [4, 16].

The scaled conjugate gradient methods are built together with safe line search based on golden section search combined with bracketing the minima [33, 34]. The implementation enables the restarting during the iteration search based on the recommendations of [21, 35]. The restarting controls the prescribed number of iterations or gradient norm. The implementation of scaled conjugate gradient uses four different updating schemes in detail described by [19, 36].

All gradient based methods apply the standard backpropagation algorithm for the estimation derivatives of the objective function with respect to weights [37]. The Levenberg-Marquardt methods approximate the Hessian matrix using first order derivatives neglecting the terms with the second order derivatives [4, 17].

2.1.3. The MLP-ANN Performance. We based the evaluation of MLP-ANN model simulations of training, testing, and validation datasets on the following statistics [26, 27, 38]:

mean absolute error (MAE)

$$\mathrm{MAE} = \frac{1}{n} \sum_{i=1}^{n} \left| R_o[i] - R_f[i] \right|, \tag{4}$$

Nash Sutcliffe efficiency (NS)

$$\mathrm{NS} = 1 - \frac{\sum_{i=1}^{n} \left(R_o[i] - R_f[i] \right)^2}{\sum_{i=1}^{n} \left(R_o[i] - \overline{R_o} \right)^2}, \tag{5}$$

fourth root mean quadrupled error (R4MS4E)

$$\mathrm{R4MS4E} = \sqrt[4]{\frac{1}{n} \sum_{i=1}^{n} \left(R_o[i] - R_f[i] \right)^4}, \tag{6}$$

persistency index (PI)

$$\mathrm{PI} = 1 - \frac{\sum_{i=1}^{n} \left(R_o[i] - R_f[i] \right)^2}{\sum_{i=1}^{n} \left(R_o[i] - R_o[i - \mathrm{LAG}] \right)^2}, \tag{7}$$

where the n represents the total number of time intervals to be predicted, the $\overline{R_o}$ is the average of observed flood runoff R_o, and LAG is the time shift describing last observed flood runoff $R_o[i - \mathrm{LAG}]$.

2.1.4. The PONS2train. The tested MLP-ANN models were implemented using the PONS2train software application. The PONS2train is software written in C++ programing language, whose main goal is to test MLP models with different architectures. The software application uses the LAPACK, BLAS, and ARMADILLO C++ linear algebra libraries [39–41]. The application is freely distributed upon a request to authors.

The PONS2train has additional features: the weight initialization can be performed using two methods. The first one follows the work of Nguyen and Widrow [42], while the second one uses random initialization coming from the uniform distribution.

Giustolisi and Laucelli [25] extensively studied the eight methods for improving the MLP performance and generalization. One of them the early stopping is incorporated in designed application. Following the recommendations of Stäger and Agarwal [43], the PONS2train also controls the avoiding of the neuron's saturation.

The important PONS2train implementation feature is the multirun and ensemble simulation. Its software design also enables further multimodel or hybrid MLP extensions [29, 44].

The software design also allows the comparative analysis of MLP's architectures with or without bias neurons in layers. The PONS2train also enables the comparison of MLP trained on shuffled and unshuffled dataset. The shuffling of data patterns follows the random permutation algorithm of Durstenfeld [45].

The MLP datasets are scaled using two methods. Both methods scale the analyses datasets into the interval $(0, k)$ with arbitrary chosen upper bound $k \leq 1$. The nonlinear scaling provides the transformed data D_{trans} obtained from original data D_{orig} using exponential transformation

$$D_{\mathrm{trans}} = \left(1 - \exp\left(-\gamma D_{\mathrm{orig}} \right) \right), \tag{8}$$

where the γ is a control parameter. The second scaling methods is a linear one.

2.2. The Dataset Description. We explored the MLP-ANN models using the rainfall and runoff time series data obtained from 10-year monitoring in the Modrava catchment 0.17 km^2. The experimental watershed was established in 1998 in upper parts of Bohemian Forest National Park. The basin belongs to the set of testbeds designed to monitor the hydrological behavior of headwater forested catchments. The watershed description shows that of Pavlasek et al. [46].

The forest cover is a clearing with young artificially planted forest combined with an undergrowth of herbs (mainly *Calamagrostis villosa*, *Avenella flexuosa*, *Scirpus sylvaticus,* and *Vaccinium myrtillus*) and bryophyte (*Polytrichastrum formosum*, *Dicranum scoparium,* and *Sphagnum girgensohnii*). A small part of the catchment (less than 10%) is covered by 40-year-old forest. The bark beetle calamity removed the original forest cover. Catchment bedrock is formed by granite, migmatite, and paragneiss covered by Haplic Podzols with depths of up to 0.9 m. The mean runoff coefficient is 0.2, mean daily runoff 1.2 mm.

Table 3: The rainfall runoff events characteristics: T_o outflow duration, Q_p flood peak, H_o runoff depth, T_r rainfall duration, I_p peak rainfall intensity, H_r rainfall depth, RC runoff coefficient, $H_{r\text{-}5D}$ antecedent rainfall depth 5 days before event, RET_e basin retention during R-R event, and RET basin retention of the whole catchment before event.

R-R event	T_o [hour]	Q_p [$m^3 \cdot s^{-1} \cdot km^{-2}$]	H_o [mm]	T_r [hour]	I_p [$mm \cdot hour^{-1}$]	H_r [mm]	RC [—]	$H_{r\text{-}5D}$ [mm]	RET_e [mm]	RET [mm]
M2_19980915-11	125	0.559	43.8	102	8.2	164	0.27	31	120.2	151.2
M2_19981027-23	100	0.902	51.8	52	9.0	128	0.4	29	76.6	105.6
M2_20010908-15	158	0.322	29.9	123	4.6	105.2	0.28	19.6	75.3	94.9
M2_20011108-11	90	0.405	22.1	45	5.8	73.6	0.3	3.2	51.5	54.7
M2_20040923-22	68	0.448	14.8	55	7.4	110.8	0.13	7.0	96.0	103.0
M2_20060527-03	77	1.093	67.2	47	12.4	156.0	0.43	21.8	88.8	110.6
M2_20070119-03	99	0.788	35.0	73	8.8	73.4	0.48	14.4	38.4	52.8
M2_20070906-18	52	0.369	14.1	52	5.6	68.6	0.21	39.6	54.5	94.1
M2_20080808-01	18	1.14	11.7	2	73.6	85.6	0.14	8.2	73.9	82.1
Mean	87.4	0.667	32.3	61.2	15.04	107.24	0.29	19.3	75.0	94.3
St. dev.	40.7	0.318	19.1	35.0	22.08	35.86	0.12	12.3	25.0	29.9

Table 4: The selected quantiles of empirical distribution functions on runoff and rainfall depths for training, testing, and validation data sets.

	Minimum	1st Quartile	Median	Mean	2nd Quartile	Maximum	St. dev.
Runoff depth [$mm \cdot hour^{-1}$]							
Training runoff	0.000	0.0154	0.0194	0.1031	0.0592	3.828	0.326
Testing runoff	0.00380	0.00760	0.0231	0.09395	0.0622	4.107	0.249
Validation runoff	0.000	0.0040	0.0270	0.1008	0.0591	3.250	0.307
Rainfall depth [$mm \cdot hour^{-1}$]							
Training rainfall	0.2	0.4	0.8	1.365	1.700	12.400	1.658
Testing rainfall	0.2	0.2	0.6	1.439	1.400	73.9	4.485
Validation rainfall	0.2	0.2	0.8	1.575	2.200	15.800	1.900

Table 5: The values of statistical measures for the benchmark SLMB model.

	Calibration dataset	Testing dataset	Validation dataset
PI [—]	0.36	0.20	0.00
NS [—]	0.96	0.82	0.96
MAE [$mm \cdot hour^{-1}$]	0.03	0.03	0.02
R4MS4E [$mm \cdot hour^{-1}$]	0.17	0.41	0.16

The most significant nine rainfall runoff events observed in hourly time step were selected from 10-year measurement. The flood runoff prediction was analyzed via proposed MLP-ANN models. The characteristics of flood events are described in Table 3. All floods events were complemented with the periods of 5 preceding days. The rainfall runoff events were divided into the nonoverlapping training, testing, and validation dataset.

The division of flood events into the datasets was made with respect to the similarity of empirical distribution functions of training, testing, and validation datasets and to their independence. The empirical distribution functions were estimated using the quantile estimation method, which was specifically developed for the description of hydrological time series (for detailed information see [47]). The selected quantiles of all datasets are shown in Table 4. The quantiles show that the distinctions of the information in training, testing, and validation datasets are not significant.

3. Results and Discussion

We tested MLP-ANN models with 4 MLP architectures; they are different according to the number of hidden layer neurons $N_{\text{hd}} = 3, 4, 5, 6$. For each MLP architecture, we prepared 11 types of MLP-ANN models according to the type of hidden layer activation function (AF) (see Table 1). Each of them was trained with 7 training algorithms (TA) (see Table 2).

All MLP-ANN datasets consisted of all available pairs of four inputs and one output. The inputs x_i were one runoff interval $Q[t-1]$ and three rainfall intervals $P[t-1]$, $P[t-2]$, and $P[t-3]$ and output R_f was formed from one runoff output $Q[t]$ for all available time intervals $[t]$. The total number of training pairs was 1270, the testing input-output datasets were 1221, and validation datasets were 1423.

Although there are suitable methodologies for selection of the proper input vector for MLP model, that is, [48–50], we based our flood forecast on small number of previous rainfall intervals and one previous runoff mainly due to fast hydrological response of analyzed watershed. The datasets were transformed using the nonlinear exponential transformation.

Table 6: The results of persistency index on analyzed optimization algorithms, the best models are marked with bold fonts. The MLP architecture with $N_{hd} = K$ labeled as 4-K-1, the best models are marked with bold font.

	ntrain	ntest	nval	PI_train	PI_test	PI_val	mPI_train	mPI_test	mPI_val
4-3-1									
FLET	298	225	111	0.67	0.52	0.14	0.33	0.24	0.07
HEST	114	91	26	0.64	0.57	0.14	0.26	0.23	0.07
PER	**1045**	**716**	**408**	0.72	0.56	**0.32**	0.40	0.26	0.08
POL	89	63	16	0.52	0.49	0.26	0.24	0.20	0.08
LM	816	511	130	**0.84**	**0.61**	0.25	**0.48**	**0.27**	**0.09**
BP	505	371	171	0.63	0.53	0.23	0.34	0.25	0.07
BP_regul	486	229	99	0.63	0.54	0.26	0.26	0.24	0.09
4-4-1									
FLET	395	292	151	0.63	0.54	0.28	0.33	0.25	0.08
HEST	186	130	21	0.64	0.49	0.16	0.25	0.21	0.06
PER	**1107**	**755**	**416**	0.75	0.60	0.22	0.43	0.27	0.09
POL	112	92	22	0.66	0.52	0.21	0.27	0.20	0.08
LM	819	550	119	**0.88**	**0.61**	**0.25**	**0.54**	**0.29**	**0.09**
BP	579	417	216	0.66	0.57	0.21	0.38	0.27	0.08
BP_regul	578	251	99	0.72	0.55	0.25	0.30	0.24	0.08
4-5-1									
FLET	413	288	157	0.77	0.52	0.18	0.33	0.25	0.08
HEST	217	165	39	0.62	0.47	0.16	0.28	0.22	0.07
PER	**1168**	**787**	**453**	0.77	0.56	0.31	0.43	0.27	0.09
POL	117	86	15	0.63	0.47	0.12	0.25	0.24	0.07
LM	859	570	91	**0.89**	**0.61**	**0.32**	**0.55**	**0.28**	**0.09**
BP	606	451	225	0.68	0.55	0.21	0.37	0.25	0.08
BP_regul	643	291	143	0.71	0.61	0.29	0.31	0.24	0.10
4-6-1									
FLET	451	342	180	0.68	0.56	0.21	0.33	0.25	0.07
HEST	218	178	42	0.66	0.51	0.14	0.26	0.21	0.06
PER	**1181**	**838**	**468**	0.82	0.59	0.23	0.43	0.28	0.08
POL	153	114	31	0.68	0.53	0.19	0.28	0.20	0.09
LM	839	579	86	**0.89**	**0.61**	0.19	**0.55**	**0.29**	0.06
BP	621	484	256	0.67	0.59	0.23	0.39	0.26	0.07
BP_regul	679	306	126	0.66	0.59	**0.24**	0.32	0.25	**0.09**

Each training algorithm was repeated 150 times. The random initialization of network weights was performed by the method of [42]. Each optimization multirun used the same values of 150 mutually different initial random vectors of weights, in order to ensure that the comparison of performances of optimization algorithms was based on similar random weights initializations.

3.1. The Benchmark Model. The flood forecast was simulated using the benchmark model based on simple linear model—SLMB. The SLMB parameters were calculated using the ordinary least squares. Table 5 shows results obtained from the simulation of SLMB benchmark model.

Since the benchmark model provides the single simulation and one value for all tested model comparison measures, we compared the results of SLMB with results of the best selected single MLP-ANN models. In model ensemble, we found MLP-ANN models, which were superior compared SLMB.

For example, the model performance based on the PI index shows all MLP-ANN provided models, which were superior compared to SLMB (see the results of Table 6). The highest differences between the best PI values of ANN and PI of SLMB were obtained on MLP-ANN trained using LM algorithm on training dataset ($PI_{ANN} - PI_{SLMB} = 0.53$). The LM and PER training algorithms provided models with the highest values of PI on testing and validation datasets ($PI_{ANN} - PI_{SLMB} = 0.41$, resp., $PI_{ANN} - PI_{SLMB} = 0.32$).

These conclusions are in agreement with the values of remaining model performance measures—MAE, NS, and R4MS4E (see Table 7). The LM and BP_regul were superior in terms of differences with SLBM according to the MAE and R4MS4E. The LM and PER were superior compared to SLMB for NS values on training, testing, and validation datasets.

Table 7: The MAE, NS, and R4M4E—trainings algorithms.

	MAE_{train}	MAE_{test}	MAE_{val}	NS_{train}	NS_{test}	NS_{val}	$R4MS4E_{train}$	$R4MS4E_{test}$	$R4MS4E_{val}$
4-3-1									
FLET	0.017	0.018	0.016	0.98	0.89	0.97	0.131	0.35	0.14
HEST	0.015	0.020	0.017	0.98	0.90	0.97	0.142	0.34	0.14
PER	0.012	0.017	0.015	0.98	0.90	**0.98**	0.124	0.34	0.12
POL	0.016	0.017	0.015	0.97	0.88	0.97	0.149	0.35	0.13
LM	**0.010**	0.015	**0.015**	**0.99**	**0.91**	0.97	**0.093**	**0.33**	0.12
BP	0.014	0.017	0.016	0.98	0.89	0.97	0.133	0.35	0.13
BP_regul	0.015	**0.014**	0.015	0.98	0.89	0.97	0.129	0.35	**0.11**
4-4-1									
FLET	0.0172	0.018	0.016	0.98	0.89	0.97	0.14	0.35	0.13
HEST	0.0154	0.020	0.016	0.98	0.88	0.97	0.15	0.35	0.14
PER	0.0125	0.017	0.015	0.98	0.91	0.97	0.13	0.34	0.13
POL	0.0138	0.019	0.015	0.98	0.89	0.97	0.15	0.35	0.13
LM	**0.0077**	**0.014**	**0.014**	**0.99**	**0.91**	0.97	**0.09**	**0.33**	**0.12**
BP	0.0140	0.018	0.016	0.98	0.90	0.97	0.13	0.35	0.13
BP_regul	0.0145	0.016	0.015	0.98	0.90	0.97	0.12	0.34	0.12
4-5-1									
FLET	0.0121	0.016	0.016	0.99	0.89	0.97	0.117	0.35	0.13
HEST	0.0159	0.020	0.017	0.98	0.88	0.97	0.137	0.35	0.12
PER	0.0129	0.015	0.015	0.99	0.90	0.98	0.115	0.34	0.12
POL	0.0159	0.022	0.018	0.98	0.88	0.97	0.143	0.35	0.14
LM	**0.0072**	0.015	**0.013**	**0.99**	**0.91**	**0.98**	**0.086**	**0.33**	**0.12**
BP	0.0139	0.016	0.016	0.98	0.90	0.97	0.122	0.35	0.13
BP_regul	0.0132	**0.014**	0.014	0.98	0.91	0.97	0.132	0.34	0.12
4-6-1									
FLET	0.0150	0.016	0.016	0.98	0.90	0.97	0.132	0.35	0.13
HEST	0.0139	0.018	0.017	0.98	0.89	0.97	0.134	0.35	0.14
PER	0.0117	0.016	0.014	0.99	0.91	0.97	0.104	0.34	0.13
POL	0.0162	0.018	0.018	0.98	0.89	0.97	0.136	0.35	0.14
LM	**0.0076**	**0.015**	**0.014**	**0.99**	**0.91**	0.97	**0.087**	**0.32**	0.14
BP	0.0139	0.017	0.015	0.98	0.90	0.97	0.131	0.35	0.14
BP_regul	0.0130	0.015	0.014	0.98	0.90	0.97	0.130	0.34	**0.12**

The similar results can be found, when comparing the results of SLMB with the best MLP-ANN models organized in terms of different transfer functions. The highest differences of PI values were on training dataset for MLP-ANN with LL transfer function ($PI_{ANN} - PI_{SLMB} = 0.48$), for testing dataset on RS transfer function ($PI_{ANN} - PI_{SLMB} = 0.41$) and for validation dataset on LL transfer function ($PI_{ANN} - PI_{SLMB} = 0.31$). These were calculated for MLP-ANN with transfer functions, which were successful in more than 10% of simulations on validation dataset.

Those results were confirmed by the values of MAE, NS, and R4MS4E obtained for the best model of a simulation ensemble. The RS transfer function provided the best results in terms of differences between$MAE_{ANN} - MAE_{SLMB} = (-0.019, -0.015, -0.005)$, $NS_{ANN} - NS_{SLMB} = (0.02, 0.09, 0.01)$, and $R4MS4E_{ANN} - R4MS4E_{SLMB} = (-0.111, -0.06, -0.03)$ on training, testing, and validation datasets.

3.2. The Optimization Algorithms. The results of MLP-ANN models were explained through the values of model performance measures, which are shown in Tables 6 and 7. All training computations controlled the neuron's saturation using the method of Stäger and Agarwal [43]. The parameters of TA (i.e., number of epochs, learning rate, etc.) were selected in such a way that the number of MLP-ANN evaluations was similar in all tested TA.

Table 6 shows the results of persistency index, which was used as a main reference index, since the PI compares the model with last observed information [38]. The best TA according to the number of successful models with PI > 0 was the PER (the scaled conjugate gradient method with Perry updating formula). The highest number of successfully trained models was found on MLP with $N_{hd} = 6$ (see the ntrained = 1181, ntest = 838, and nval = 468 in Table 6).

When comparing the performance of TA according to the best single value of PI (see columns PI_train, PI_test,

Table 8: The results of persistency index on tested transfer functions, the best models are marked with bold font.

	ntrain	ntest	nval	PI_train	PI_test	PI_val	mPI_train	mPI_test	mPI_val
4-3-1									
CL	275	225	110	0.80	0.46	0.24	0.35	0.24	**0.08**
CLm	555	381	**207**	0.70	**0.56**	0.26	0.37	0.25	0.08
HT	489	283	148	0.72	0.56	**0.27**	**0.42**	0.26	0.08
LL	355	273	139	**0.84**	0.55	0.21	0.35	0.25	0.08
RS	**566**	**417**	198	0.73	0.56	0.22	0.40	**0.27**	0.08
4-4-1									
CL	290	241	129	0.74	0.52	0.17	0.35	0.27	0.08
CLm	**615**	402	**237**	0.75	0.56	0.24	0.38	0.27	0.08
HT	578	351	152	0.75	0.56	0.25	**0.44**	0.27	0.08
LL	384	300	157	**0.83**	0.57	**0.28**	0.35	0.26	**0.09**
RS	608	**475**	209	0.75	**0.61**	0.25	0.42	**0.28**	0.08
4-5-1									
CL	311	256	144	0.77	0.53	0.24	0.35	0.26	0.09
CLm	632	437	**245**	0.70	0.58	0.21	0.39	0.26	0.08
HT	574	321	147	0.75	0.61	0.31	**0.45**	0.27	**0.09**
LL	432	337	187	**0.79**	0.56	**0.31**	0.35	0.25	0.08
LS	377	291	115	0.74	0.57	0.21	0.35	0.25	0.08
RS	**659**	**517**	242	0.74	**0.61**	0.29	0.42	**0.27**	0.09
4-6-1									
CL	319	269	145	0.77	0.56	0.21	0.34	0.27	**0.08**
CLm	654	471	259	0.69	0.59	**0.24**	0.39	0.27	0.07
HT	601	361	162	0.71	0.59	0.20	**0.45**	0.27	0.08
LL	437	358	185	**0.75**	0.53	0.17	0.35	0.27	0.08
LS	391	324	114	0.71	0.55	0.20	0.35	0.22	0.07
RS	**651**	**539**	**263**	0.72	**0.61**	0.22	0.42	**0.28**	0.08

and PI_val in Table 6) and the average performance of best MLP-ANN models on PI (see columns mPI_train, mPI_test, and mPI_val in Table 6), the Levenberg-Marquardt algorithm was mostly superior compared to all remaining TA, except for three cases, when the PER and BP_regul were better on validation datasets for MLP with $N_{hd} = 3, 6$ on best single value of PI and for average of mPI_val for $N_{hd} = 6$.

Table 7 displays the results of best models for remaining statistical measures of MLP-ANN models trained on tested TA. Only three algorithms were superior at least for one architecture of MLP and on one dataset. They are LM, PER, and BP_regul. Again, the LM was mostly superior compared to the other tested TA. The differences between results of LM and PER and BP_regul were very small.

The best values of NS were in agreement with values of PI (see, e.g., the PER on MLP with $N_{hd} = 3$). The BP_regul was better in terms of the length of residuals for MAE_test on MLP ANN models with $N_{hd} = 3, 5$. Also when comparing the simulation of peak flow in terms of R4MS4E, the BP_regul was better on MLP with $N_{hd} = 3, 6$ for validation dataset.

Our finding are in agreement with results on runoff forecast of Piotrowski and Napiorkowski [18], who compared the Levenberg-Marquardt approach even with more robust global optimization schemes, and found that the LM provides comparable results with MLP trained using the selected evolutionary computation methods.

3.3. The Transfer Functions. The results of PI, MAE, NS, and R4MS4E are shown in Tables 8 and 9. The PI has again served as a reference. We trained the MLP with all AF listed in Table 1. Tables 8 and 9 show the results of AF for MLP-ANN models, which were successful in more than 10% of simulations on validation dataset.

When comparing the absolute values of number of MLP-ANN models with PI $>$ 0, the models with two AF (RS and CLm) were superior compared to MLP models with remaining 9 AFs. The MLP with RS provided the larger number of better models in terms of PI value on 8 datasets, while the MLP with CLm transfer function was successful on 4 datasets.

RS was also the most successful TA on training dataset at MLPs with $N_{hd} = 4, 5, 6$ (note that for $N_{hd} = 3$ the differences in PI between RS and CLm are almost insignificant). The LL also provided good results on training dataset (for all tested values of N_{hd}) and on validation data for $N_{hd} = 4, 5$.

The mean performances based on arithmetical means of PI values of best models showed that three AFs were superior compared to remaining 8 AFs (see mPI_train, mPI_test, and

Table 9: The MAE, NS, and R4MS4E on activation functions, the best models are marked with bold font.

	MAE_{train}	MAE_{test}	MAE_{val}	NS_{train}	NS_{test}	NS_{val}	$R4MS4E_{train}$	$R4MS4E_{test}$	$R4MS4E_{val}$
4-3-1									
CL	0.10	0.10	0.10	0.72	0.52	0.68	0.40	0.63	0.42
CLm	0.06	0.06	0.06	0.82	**0.65**	**0.81**	0.34	0.71	0.36
HT	0.06	0.06	0.06	0.81	0.60	0.79	0.34	0.74	0.37
LL	0.08	0.08	0.08	0.76	0.57	0.73	0.34	0.64	0.37
RS	**0.06**	**0.06**	**0.05**	**0.82**	0.65	0.80	**0.31**	**0.61**	**0.34**
4-4-1									
CL	0.10	0.10	0.10	0.71	0.53	0.68	0.40	0.62	0.41
CLm	0.06	0.06	0.05	**0.83**	0.66	**0.82**	0.30	0.67	0.33
HT	0.06	0.06	0.06	0.83	0.63	0.81	0.32	0.71	0.36
LL	0.08	0.08	0.07	0.79	0.59	0.75	0.31	0.62	0.35
RS	**0.05**	**0.05**	**0.05**	0.82	**0.67**	0.81	**0.29**	**0.59**	**0.32**
4-5-1									
CL	0.10	0.10	0.10	0.72	0.56	0.68	0.41	0.62	0.42
CLm	0.05	0.05	0.05	0.84	0.68	**0.84**	0.29	0.69	0.32
HT	0.05	0.05	0.05	**0.84**	0.64	0.83	0.29	0.70	0.33
LL	0.07	0.07	0.07	0.82	0.61	0.78	0.30	0.59	0.33
LS	0.07	0.07	0.06	0.82	0.60	0.79	0.28	**0.56**	0.31
RS	**0.05**	**0.05**	**0.05**	0.84	**0.69**	0.83	**0.28**	0.58	**0.30**
4-6-1									
CL	0.10	0.10	0.10	0.75	0.59	0.69	0.42	0.60	0.42
CLm	**0.05**	**0.05**	**0.05**	0.84	0.69	0.83	0.28	0.65	0.31
HT	0.05	0.05	0.05	**0.85**	0.65	**0.83**	0.28	0.65	0.32
LL	0.07	0.07	0.07	0.81	0.62	0.78	0.29	0.57	0.32
LS	0.07	0.07	0.06	0.82	0.60	0.78	**0.28**	**0.55**	0.31
RS	0.05	0.05	0.05	0.84	**0.70**	0.82	0.28	0.55	**0.30**

mPI_val in Table 8). They were CL, HT, and RS MLP ANN models. Their differences of PI were again very small.

Table 9 shows the averages of MAE, NS, and R4MS4E on set of tested models. The results point out that the RS transfer function provided in summary superior values compared to rest of tested AF. The CLm, HT, and LS activation functions were on some datasets better in terms of mean values of tested statistical measures but the differences between the RS MLP ANN models were again negligible.

When reflecting the results of da S. Gomes et al. [15], who recommended the CL, CLm, and LL functions on MLP ANN models, we point out the ability of the MLP models with RS to improve the flood runoff forecast.

Our findings on the selection of suitable AF on MLP ANN models recommend that different AF should be tested during the implementation of MLP models for flood runoff forecast.

4. Conclusions

During the extensive computational test, we trained in total the 46200 models of multilayer perceptron with one hidden layer. The main aim of computational exercise was the evaluation of the impacts of the transfer function selection and the test of selected local optimization schemes on flood runoff forecast.

Using the rainfall runoff data of nine of the most significant flood events, we analyzed the short term runoff forecast on small watershed with fast hydrological response. The developed MLP ANN models were able to predict flood runoff using the records of past rainfall and runoff from the basin.

When comparing the tested MLP ANN models with benchmark simple linear model, the developed MLP models were superior in terms of values of model performance measures compared to the SLMB.

The PONS2Train software application was developed for the purposes of the evaluation of MLP-ANN models with different architectures and for providing the simulations of neural network flood forecast.

When analyzing the 7 different gradient oriented optimization schemes we found that the Levenberg-Marquardt algorithm was superior compared to the tested set of scaled conjugate gradient methods and two first order local optimization schemes.

When analyzing the 11 different transfer functions used in hidden neurons we found that the RootSig function was according to the values of four model performance measures

most promising activation function in terms of flood runoff forecast.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] H. R. Maier and G. C. Dandy, "Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications," *Environmental Modelling and Software*, vol. 15, no. 1, pp. 101–124, 2000.

[2] C. W. Dawson and R. L. Wilby, "Hydrological modelling using artificial neural networks," *Progress in Physical Geography*, vol. 25, no. 1, pp. 80–108, 2001.

[3] H. R. Maier, A. Jain, G. C. Dandy, and K. P. Sudheer, "Methods used for the development of neural networks for the prediction of water resource variables in river systems: current status and future directions," *Environmental Modelling and Software*, vol. 25, no. 8, pp. 891–909, 2010.

[4] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, NY, USA, 1995.

[5] R. D. Reed and R. J. Marks, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, MIT Press, Cambridge, Mass, USA, 1998.

[6] A. W. Minns and M. J. Hall, "Artificial neural networks as rainfall-runoff models," *Hydrological Sciences Journal*, vol. 41, no. 3, pp. 399–417, 1996.

[7] H. K. Cigizoglu, "Estimation, forecasting and extrapolation of river flows by artificial neural networks," *Hydrological Sciences Journal*, vol. 48, no. 3, pp. 349–361, 2003.

[8] N. J. de Vos and T. H. M. Rientjes, "Constraints of artificial neural networks for rainfall-runoff modelling: trade-offs in hydrological state representation and model evaluation," *Hydrology and Earth System Sciences*, vol. 9, no. 1-2, pp. 111–126, 2005.

[9] G. Napolitano, F. Serinaldi, and L. See, "Impact of EMD decomposition and random initialisation of weights in ANN hindcasting of daily stream flow series: an empirical examination," *Journal of Hydrology*, vol. 406, no. 3-4, pp. 199–214, 2011.

[10] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[11] N. J. de Vos and T. H. M. Rientjes, "Multiobjective training of artificial neural networks for rainfall-runoff modeling," *Water Resources Research*, vol. 44, no. 8, 2008.

[12] E. Toth and A. Brath, "Multistep ahead streamflow forecasting: role of calibration data in conceptual and neural network modeling," *Water Resources Research*, vol. 43, no. 11, 2007.

[13] M. P. Rajurkar, U. C. Kothyari, and U. C. Chaube, "Modeling of the daily rainfall-runoff relationship with artificial neural network," *Journal of Hydrology*, vol. 285, no. 1–4, pp. 96–113, 2004.

[14] C. M. Zealand, D. H. Burn, and S. P. Simonovic, "Short term streamflow forecasting using artificial neural networks," *Journal of Hydrology*, vol. 214, no. 1–4, pp. 32–48, 1999.

[15] G. S. da S.Gomes, T. B. Ludermir, and L. M. M. R. Lima, "Comparison of new activation functions in neural network for forecasting financial time series," *Neural Computing & Applications*, vol. 20, no. 3, pp. 417–439, 2011.

[16] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, New York, NY, USA, 2003.

[17] M. Hagan and M. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[18] A. P. Piotrowski and J. J. Napiorkowski, "Optimizing neural networks for river flow forecasting—evolutionary computation methods versus the levenberg-marquardt approach," *Journal of Hydrology*, vol. 407, no. 1–4, pp. 12–27, 2011.

[19] A. E. Kostopoulos and T. N. Grapsa, "Self-scaled conjugate gradient training algorithms," *Neurocomputing*, vol. 72, no. 13–15, pp. 3000–3019, 2009.

[20] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA Journal of Numerical Analysis*, vol. 8, no. 1, pp. 141–148, 1988.

[21] M. J. D. Powell, "Restart procedures for the conjugate gradient method," *Mathematical Programming*, vol. 12, no. 2, pp. 241–254, 1977.

[22] A. Shamseldin, A. Nasr, and K. O'Connor, "Comparsion of different forms of the multi-layer feed-forward neural network method used for river flow forecasting," *Hydrology and Earth System Sciences*, vol. 6, no. 4, pp. 671–684, 2002.

[23] R. R. Shrestha, S. Theobald, and F. Nestmann, "Simulation of flood flow in a river system using artificial neural networks," *Hydrology and Earth System Sciences*, vol. 9, no. 4, pp. 313–321, 2005.

[24] H. Yonaba, F. Anctil, and V. Fortin, "Comparing sigmoid transfer functions for neural network multistep ahead streamflow forecasting," *Journal of Hydrologic Engineering*, vol. 15, no. 4, pp. 275–283, 2010.

[25] O. Giustolisi and D. Laucelli, "Improving generalization of artificial neural networks in rainfall-runoff modelling," *Hydrological Sciences Journal*, vol. 50, no. 3, pp. 439–457, 2005.

[26] C. W. Dawson, R. J. Abrahart, and L. M. See, "HydroTest: further development of a web resource for the standardised assessment of hydrological models," *Environmental Modelling & Software*, vol. 25, no. 11, pp. 1481–1482, 2010.

[27] C. W. Dawson, R. J. Abrahart, and L. M. See, "HydroTest: a web-based toolbox of evaluation metrics for the standardised assessment of hydrological forecasts," *Environmental Modelling and Software*, vol. 22, no. 7, pp. 1034–1052, 2007.

[28] A. Y. Shamseldin, "Application of a neural network technique to rainfall-runoff modelling," *Journal of Hydrology*, vol. 199, no. 3-4, pp. 272–294, 1997.

[29] W. Wang, P. H. A. J. M. V. Gelder, J. K. Vrijling, and J. Ma, "Forecasting daily streamflow using hybrid ANN models," *Journal of Hydrology*, vol. 324, no. 1–4, pp. 383–399, 2006.

[30] B. Pang, S. Guo, L. Xiong, and C. Li, "A nonlinear perturbation model based on artificial neural network," *Journal of Hydrology*, vol. 333, no. 2–4, pp. 504–516, 2007.

[31] A. P. Piotrowski and J. J. Napiorkowski, "A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling," *Journal of Hydrology*, vol. 476, pp. 97–111, 2013.

[32] C. Imrie, S. Durucan, and A. Korre, "River flow prediction using artificial neural networks: generalisation beyond the calibration range," *Journal of Hydrology*, vol. 233, no. 1–4, pp. 138–153, 2000.

[33] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C++: The Art of Scientific Computing*, Cambridge University Press, 2002.

[34] T. Masters, *Practical Neural Network Recipes in C++*, Morgan Kaufmann, 1st edition, 1993.

[35] N. Andrei, "Scaled conjugate gradient algorithms for unconstrained optimization," *Computational Optimization and Applications. An International Journal*, vol. 38, no. 3, pp. 401–416, 2007.

[36] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, pp. 149–154, 1964.

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[38] P. K. Kitanidis and R. L. Bras, "Real-time forecasting with a conceptual hydrologic model. 2: applications and results," *Water Resources Research*, vol. 16, no. 6, pp. 1034–1044, 1980.

[39] E. Anderson, Z. Bai, J. Dongarra et al., "Lapack: a portable linear algebra library for high-performance computers," in *Proceedings of the ACM/IEEE conference on Super- computing*, pp. 2–11, IEEE Computer Society Press, Los Alamitos, Calif, USA, November 1990.

[40] L. S. Blackford, J. Demmel, J. Dongarra et al., "An updated set of basic linear algebra subprograms (BLAS)," *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 135–151, 2002.

[41] C. Sanderson, "Armadillo: an open source C++ linear algebra library for fast prototyping and computationally intensive experiments," Tech. Rep., NICTA, Sydney, Australia, 2010.

[42] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of adaptive weights," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '90)*, vol. 1–3, pp. C21–C26, International Neural Network Society, San Diego, Calif, USA, June 1990.

[43] F. Stäger and M. Agarwal, "Three methods to speed up the training of feedforward and feedback perceptrons," *Neural Networks*, vol. 10, no. 8, pp. 1435–1443, 1997.

[44] Z. Huo, S. Feng, S. Kang, G. Huang, F. Wang, and P. Guo, "Integrated neural networks for monthly river flow estimation in arid inland basin of Northwest China," *Journal of Hydrology*, vol. 420-421, pp. 159–170, 2012.

[45] R. Durstenfeld, "Algorithm 235: random permutation," *Communications of the ACM*, vol. 7, no. 7, p. 420, 1964.

[46] J. Pavlasek, M. Tesar, P. Maca et al., "Ten years of hydrological monitoring in upland microcatchments in the bohemian forest, Czech Republic," in *Status and Perspectives of Hydrology in Small Basins*, pp. 213–219, IAHS, 2010.

[47] R. J. Hyndman and Y. Fan, "Sample quantiles in statistical packages," *American Statistician*, vol. 50, no. 4, pp. 361–365, 1996.

[48] G. J. Bowden, G. C. Dandy, and H. R. Maier, "Input determination for neural network models in water resources applications. Part 1: background and methodology," *Journal of Hydrology*, vol. 301, no. 1–4, pp. 75–92, 2005.

[49] R. J. May, H. R. Maier, G. C. Dandy, and T. M. K. G. Fernando, "Non-linear variable selection for artificial neural networks using partial mutual information," *Environmental Modelling and Software*, vol. 23, no. 10-11, pp. 1312–1326, 2008.

[50] R. J. May, G. C. Dandy, H. R. Maier, and J. B. Nixon, "Application of partial mutual information variable selection to ANN forecasting of water quality in water distribution systems," *Environmental Modelling & Software*, vol. 23, no. 10-11, pp. 1289–1299, 2008.